Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA?

Yevgeniy Dodis* Prashant Puniya†

Abstract

Cascade chaining is a very efficient and popular mode of operation for building various kinds of cryptographic hash functions. In particular, it is the basis of the most heavily utilized SHA function family. Recently, many researchers pointed out various practical and theoretical deficiencies of this mode, which resulted in a renewed interest in building specialized modes of operations and new hash functions with better security. Unfortunately, it appears unlikely that a new hash function (say, based on a new mode of operation) would be widely adopted before being standardized, which is not expected to happen in the foreseeable future.

Instead, it seems likely that practitioners would continue to use the cascade chaining, and the SHA family in particular, and try to work around the deficiencies mentioned above. In this paper we provide a thorough treatment of how to soundly design a secure hash function H' from a given cascade-based hash function H for various cryptographic applications, such as collision-resistance, one-wayness, pseudorandomness, etc. We require each proposed construction of H' to satisfy the following "axioms".

1. The construction should consist of one or two *"black-box" calls* to H.
2. In particular, one is not allowed to know/use anything about the internals of H, such as modifying the initialization vector or affecting the value of the chaining variable.
3. The construction should support variable-length inputs.
4. Compared to a single evaluation of $H(M)$, the evaluation of $H'(M)$ should make at most a fixed (small constant) number of extra calls to the underlying compression function of H. In other words, the efficiency of H' is negligibly close to that of H.

We discuss several popular modes of operation satisfying the above axioms. For each such mode and for each given desired security requirement, we discuss the weakest requirement on the compression function of H which would make this mode secure. We also give the implications of these results for using existing hash functions SHA-x, where $x \in \{1, 224, 256, 384, 512\}$.

*Email: dodis@cs.nyu.edu

†Email: puniya@cs.nyu.edu

1 Introduction

The *Cascade construction* is a very elegant way to build a hash function H on arbitrary-length inputs from a given compression function h on fixed-length inputs. Recall that for a given $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, one can define a hash function H, parametrized by an initialization vector $IV \in \{0,1\}^n$, as follows (where input $x = x_1 \parallel \ldots \parallel x_\ell$ and $x_i \in \{0,1\}^\kappa$ for $i = 1 \ldots \ell$):

$$H(x_1 \parallel \ldots \parallel x_\ell) = h(x_\ell, h(\ldots, h(x_1, IV)\ldots))$$

We will refer to this as the MD mode or the plain MD mode (after Merkle-Damgård). The most abundant use of the MD mode in practice comes in the design of the industry-standard hash family SHA (which consists of several specific hash functions SHA-x, where $x \in \{1, 224, 256, 384, 512\}$). Unfortunately, despite its elegance and simplicity, the plain MD mode has several deficiencies. For instance, it does not guarantee that a "global" collision of H implies a "local" collision of the compression function h, unless one preprocesses the input into a suffix-free form before applying H [10] (as we already mentioned, the particular suffix-free encoding of appending the message length is called *MD strengthening*, and is actually used in the SHA family for this reason). More seriously, it was shown by Coron et al. [9] that even MD strengthening falls prey to the "extension attack" [1] which makes it insufficient for domain extension of random oracle. Moreover, this deficiency disqualifies the natural use of "plain MD" in the design of "pseudorandom functions" [3]. Other problems also arise when the MD mode is used in applications such as key derivation [11] and target collision-resistance (or UOWHFs [2]) [5, 24].

Apart from the issues mentioned above, several other deficiencies of the MD mode against exponential-time attacks have been discovered [14, 16]. All these deficiencies, coupled with the improved brute-force attacks on the popular SHA-1 hash function proposed recently [25, 26], suggest that it is time to design a better, more "secure" mode of operation for building a variable-length input hash function. With this purpose, NIST has been organizing several workshops dedicated to coming up with the next generation hash functions [21]. However, this process will take some time, and it does not appear that such hash functions would be standardized and widely accepted in any foreseeable future. Therefore, practitioners are "stuck" with the prospect of using existing hash functions, despite all their deficiencies. Hence, there is a pressing need to design immediate "fixes" to the MD paradigm, without changing it drastically.

There are two aims in coming up with such "fixes" to the MD mode. The first, and so far the most popular, aim is to design a slight variant of the MD mode that provably preserves a given security property of the compression function, and to do so in the most aesthetic and efficient manner. We mention only a few of the many examples of this approach. For collision-resistance, we already mentioned the well known technique of *MD strengthening*. For another example, by viewing the initialization vector as the key and applying a *prefix-free encoding* to the message, one can obtain a variable-length input pseudorandom function from a fixed-length input pseudorandom compression function [3]. In the case of target collision-resistance, Shoup [24] designed an elegant mode for building target collision-resistant (TCR) hash functions (or UOWHFs [22]) from a TCR compression function by cleverly XORing certain masks to the internal chaining variables in the MD construction. The common feature in all these results is that one assumes *exactly the same* property from the compression function h as the desired property from the hash function H. In many cases, such as the PRF and TCR examples, this means that a "secure" mode must be sufficiently different from the plain MD so that its implementation requires a non-trivial modification to the SHA implementation. Concretely, the SHA family uses a fixed public IV (as opposed to arbitrary secret IV needed for PRFs), while in the TCR case one cannot XOR the corresponding masks without modifying the internals of SHA.

Of course, we are not saying that the required modifications are too "complicated" to be correctly implemented by a serious programmer. In fact, they are not. Our point is that, irrespective of simplicity and conceptual similarity to the existing implementations, they require one to tinker with the internals of such standard implementations. And this

[1] I.e., given $H(x)$ and any extension y, one can compute $H(x \parallel y)$ without knowing x.

[2] Which stands for Universal One-Way Hash Functions.

is not only error-prone and requiring low-level programming (which could result in less optimized implementations than those done by the experts), but goes against the whole philosophy of modular design. We do not want our security engineers to know all the low-level cryptographic details. Instead, they should understand the higher-level picture of the protocols they are trying to build, and never need to worry about existing low-level libraries.

This brings us to the second approach, where one explicitly aims to design a "secure" mode that uses only *black-box calls* to the plain MD mode.[3] For instance, MD strengthening satisfies this property. Other important examples include the HMAC mode for pseudorandom functions [3] and the results for domain extension of random oracle in [9]. The attractive feature of these results is that they result in a hash function with the desired property without tinkering with the internals of SHA, and can use any off-the-shelf implementation. Moreover, all these examples also satisfy the *property-preserving* property described above, and do so without any noticeable efficiency penalties as compared to the solutions following the first approach. Concretely, at the price of one or two (or sometimes zero!) extra calls to the compression function h — which is negligible for all practical purposes —, one manages to achieve the desired goal without tinkering with the internals of the existing hash functions.

OUR GOAL. Not surprisingly, we will emphasize the latter approach in coming up with "fixes" for existing hash functions. That is, we consider the question of building a hash function H' achieving a given security property P using a black-box MD-based hash function H (with an unknown compression function h). We require that the proposed construction H' satisfies the following "axioms":

1. The construction should consist of one or two *"black-box" calls* to H. In particular, the construction is not allowed to use any knowledge of or tinker with the internals of the hash function H.
2. The construction must support variable-length inputs.
3. Compared to a single evaluation of $H(M)$, the evaluation of $H'(M)$ should make at most a fixed (small constant) number of extra calls to the underlying compression function of H. In other words, the efficiency of H' is negligibly close to that of H.

The motivation behind requiring the construction H' to satisfy these axioms is from the viewpoint of a practitioner who understands the properties of the hash function that are needed for the security of his cryptosystem, but who wants to use an off-the-shelf standardized hash function implementation without tinkering with its internals. Such a practitioner would be willing to sacrifice the *property-preserving* aspect of the "fix" in favor of a black-box implementation.

In fact, the above "axioms" leave very little freedom in choosing the modes of operation for H'. The resulting modes are essentially the *most widely-utilized* constructions appearing in practical implementations:

1. *Plain MD Construction:* This captures the notion that the application uses the hash function as it is. We will denote this mode of operation as H.
2. *Encode-then-MD Construction:* In this case, the user encodes the hash function input before applying the plain MD construction. Examples of popular encoding schemes used are suffix-free encoding and prefix-free encoding. We will refer to the corresponding constructions as the *prefix-free MD construction* H_{pre} and the *suffix-free MD construction* H_{suf}.
3. *MD-then-Chop Construction:* Here the user applies the plain MD mode and only uses part of the output while discarding the remaining bits. In particular, existing hash functions SHA-224 and SHA-384 are obtained this way from SHA-256 and SHA-512, respectively. We denote the MD-then-chop construction that chops s bits of the output as H_{chop_s}.
4. *NMAC/HMAC Construction:* The version of the NMAC construction that we consider simply composes two applications of the plain MD mode with possibly different initialization vectors IV_1 and IV_2. While not obeying the first axiom, the NMAC construction serves as a nice abstraction for the HMAC construction which does satisfy all our axioms (but is slightly harder to formally analyze in some cases). Concretely, the HMAC

[3]In practice, with MD strengthening, but we ignore this aspect for now.

construction uses the NMAC construction with $IV_1 = h(IV, \alpha_1) = H(\alpha_1)$ and $IV_2 = h(IV, \alpha_2) = H(\alpha_2)$, where each α_i is either the null string $\perp$ (in which case we let $h(IV, \perp) = IV$) or a single κ-bit block. We denote the NMAC construction as H_{nmac} and the HMAC construction as H_{hmac}.

Now we can finally rephrase our goal as follows. Given a particular desired security property P (such as collision-resistance or pseudorandomness) and one of the 4 modes of operation above (which all satisfy our axioms), find the weakest security assumption(s) P' on the compression function h which would make the corresponding mode satisfy P (or determine that the construction is insecure for any h). Ideally, this security property P' for h would be P itself (which would result in a *property-preserving mode of operation*). However, unlike most previous work, property preservation is not our primary concern. In particular, we will not declare a mode of operation to be "insecure" for a property P simply because it is not property-preserving for P. Instead, we will find the weakest security property P' of the compression function that makes the resulting construction secure. This will allow the practitioners to decide whether or not it is reasonable to assume that the compression function of existing hash functions, such as SHA, satisfy the property P', even if P' is (slightly) stronger than P.

OUR RESULTS. We achieve our main goal for a very wide variety of security properties including *collision-resistance (CR)*, *pseudorandomness (PR)*, *indifferentiability from random oracle (RO)*, *message authentication (MAC)*, *target collision-resistance (TCR)*, *second preimage-resistance (SPR)*, *randomness extraction (RE)* and *one-wayness (OW)*. In each case, and for each of the four popular modes above, we will identify the needed property P' on h. In some cases, the needed P' easily follows from some existing work (for instance, from [9] in the case of domain extension of random oracle). In other cases, it required some minor, but important modifications to the existing results in order to satisfy our axioms. For example, by assuming that "$h(IV, random) = random$" in addition to h being a PRF when keyed with the first n bits of its input, we could build a variable length PRF using the encode-then-MD mode and adjusting the proof of [3]. More interestingly, by making extra assumptions on h, in some cases we can prove security of the modes which were previously believed "insecure" because they were not property-preserving. Finally, in some cases the proof will involve careful and non-trivial modification of previous results. For example, this is the case when analyzing the one-wayness of the H_{suf} construction.

In addition to giving an exhaustive "mode × property" guide (see figure 1) for achieving a given security property with a given popular mode, in each section we also mention the practical implication of our results when using existing hash functions SHA-x, where $x \in \{1, 224, 256, 384, 512\}$.

RELATED WORK. We have already cited many of the relevant papers. In particular, the variants of the MD mode that are useful in the property-preservation of collision-resistance [10], pseudorandomness [3, 4], message-authentication [1, 20], random oracles [9] and randomness extraction [11]. We also mention the works of [7, 8] concerned with multiple property-preservation; namely, designing a single mode of operation which simultaneously preserves several properties. Unfortunately, the modes of [7, 8] do not satisfy our axioms. Finally, we mention the work of Halevi and Krawczyk [13], which concentrated on building TCR hash functions, and is the closest in spirit to our motivation (indeed, we will use their results when discussing the TCR property). The authors built TCR hash functions using the encode-then-MD mode, and showed a simple coding scheme that yields a secure TCR hash function under an appropriately strong assumption on the underlying compression function h (still weaker than CR, but stronger than TCR).

LOCATION OF THE KEY IN KEYED CONSTRUCTIONS. We note that for keyed constructions, such as constructions of pseudorandom and TCR functions, there are more than one possibilities for each hash function mode of operation. In particular, any construction for these primitives must specify the location of the key. In keeping with the black-box nature of the modes of operation, we prevent popular keying methods such as setting the key to be the IV or XORing the key into the chaining variables since this violates our basic axioms.

Moreover, we also do not consider the dedicated-key setting [1, 8], where there is separate space for the key in each application of the compression function. This is because existing hash functions do not support such dedicated keys. Even though we may consider the key to be part of the message block bits, we do not analyze this method since it yields constructions with poor input bandwidth (thus violating our last axiom). Hence, we will only consider modes of operation which incur an additive constant overhead compared to the plain MD mode.

	Plain MD	Encode-then-MD	MD-then-Chop	NMAC/HMAC
CRHF	(1) + (2)	Suf-Free+(1) Pre-Free+(1)+(2)	(1') + (2)	N/HMAC+(1)+(2) $\alpha_1 \neq \perp$
PRF	Append key + (1)+(2)+(4)	SF+(1)+(4) (append) PF+(2')+(3) (prepend)	Prepend key + (2')+(3')	N/H+(3)+(4) (prepend) Any $IVs/\alpha s$
RO	Not Secure	Suf-Free not secure Pre-Free+(5)	(5) worse security	NMAC/HMAC+(5) $IV_1 \neq IV_2$; $\alpha_1 \neq \alpha_2$
MAC	Append key + (1)+(2)+(6)	SF+(1)+(6) (append) PF+(1)+(2)+(6)(app.)	Append key + (1)+(2)+(6')	N/H+(1)+(2)+(6) Any IVs/αs
TCR	$key \oplus blks$ (7) + (9)	SF+(7) ($key \oplus blks$) PF+(7)+(9)	$key \oplus blks$ (7') + (9)	N/H+(7)+(9) (append) Any IVs/αs ($key \oplus blks$)
SPR	(8) + (9)	SF+(9) PF+(8)+(9)	(8') + (9)	N/H+(8)+(9) Any IVs/αs
RExt	(10) $H_\infty(M) \wedge H_\infty(m_\ell)$	MDS + (10)(SF/PF??) $H_\infty(M) \wedge H_\infty(m_\ell)$	(10) $H_\infty(M)$	NMAC + (10) HMAC??
OWF	(2)+(11)	MDS+(2)+(11) (SF/PF??)	(2')+(11)	NMAC+(2)+(11) HMAC??

Assumptions on compression function:

(1)=Collision Resistance (CR)	(1')=CR after Chop
(2)=Output Regular	(2')=$h(U_n, \cdot)$ is output regular
(3)=standard PRF (sPRF)	(3')=sPRF after Chop
(4)=dual PRF (dPRF)	
(5)=FIL-RO	
(6)=MAC with κ-bit key	
(7)=enhanced SPR (eSPR)	(7')=eSPR after Chop
(8)=computed SPR (cSPR)	(8')=cSPR after Chop
(9)=Fixed-point at random IV	
(10)=Family of random functions	
(11)=One-way function	

Misc.

SF=Suffix-free
PF=Prefix-free
MDS=MD Strengtheining
??=not known to be secure
RExt=Randomness Extrn.
$Key \oplus Blks$ =XOR key to each block

Figure 1: Table for comparing Security Property vs. Mode of operation

ARE WE ASKING TOO MUCH? In our motivation, we advocated the fact that the security officers should not know (or worry about) the low-level details of the hash function implementations. In particular, we do not want them to manually modify the internals of SHA. On the other hand, to use our result they have to be "smart enough" to understand the purpose of their application of the hash function, so they can use our black-box workarounds. For example, they need to know if H' is used for collision-resistance, key derivation, one-wayness, etc. Aren't we asking too much? Should not the security engineer just believe that the existing hash function will be "magically applicable" for whatever intuitive use (s)he has in mind (therefore making this paper "useless")?

We give two answers. First, we personally believe that a person designing a cryptographic protocol using a hash function *should* know what security properties this hash function should satisfy. (And this does not contradict our desire to protect them from low-level details!) Second, in order for the security engineer to use a hash function in the "magical" way above, the function should not have the weaknesses of the SHA family we mentioned earlier. Thus, until a new, "magic" hash function is built and standardized, we simply *cannot achieve* a positive answer to our question, even if we *want* our engineers to be "dumb" and not understanding what they is doing (which we personally disagree with)! Until then, we believe that the results of this paper are meaningful and useful.

2 Security of MD modes

We will analyze each of the security properties that actual hash functions are often required to satisfy, and find the minimal assumptions on the compression function that are necessary to prove the security of each of the black-box modes of operation for this security notion. As we discussed, we will not restrict ourselves to the case of property-preservation and in some cases, we will need to make slightly stronger assumptions on the compression function than the security notion desired.

Since the focus of our paper is mostly qualitative, in terms of when (i.e. for which applications) does it make more sense to use some particular mode of operation, so we will keep the discussion "slightly informal" by using more asymptotic definitions for the security notions. We assume basic familiarity with these notions, but provide the formal definitions in the appendix A if one needs. Due to space constraints, we only give the security of the modes of operation for collision-resistance, pseudorandomness, target collision-resistance and one-wayness in the main body. The discussion for other security notions can be found in appendices B-E.

2.1 Collision Resistance

We will analyze each of the four modes for minimal assumptions required on the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ needed in order to prove its collision resistance. A construction will be called ϵ collision resistant if the maximum advantage of an efficient attacker in finding a collision is ϵ. As we discussed, in some cases, the security property needed for the compression function h may be stronger than collision resistance.

PLAIN MD CONSTRUCTION. It is a well-known fact that simply assuming collision resistance of the compression function does not suffice to prove collision resistance of the plain MD construction. Indeed, if the compression function h has a *fixed-point* such that there is some $x \in \{0,1\}^\kappa$ such that: $h(x, IV) = IV$. Then the output of the plain MD construction H collides for the inputs x and $x \parallel m$, for any m. Thus we, at least, need the compression function to satisfy the following property.

Assumption 1 (No Fixed-Points) *A function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a ϵ* secure *against fixed points if for a randomly chosen $IV \in \{0,1\}^n$ no efficient machine A has success probability more than ϵ of finding a sequence of κ-bit blocks $x_1 \ldots x_i$ such that,*

$$h(x_i, h(\ldots, h(x_1, IV)\ldots)) = IV$$

If the compression function is such that no efficient attacker can find such fixed points (along with being collision resistant), then the plain MD construction can be proven to be collision resistant.

Observation 1 *The plain MD construction can be proven to be collision resistant if the compression function is collision resistant and is secure against fixed-points. The proof of collision resistance for this case works as in [10],*

The *no fixed-points* assumption allows us to prove collision resistance of the plain MD construction, but it is a non-standard assumption and it is not intuitively clear as to which compression functions satisfy this property. But since we are already assuming the compression function to be collision-resistant, perhaps we can prove this result by making a weaker and cleaner additional assumption on the compression function. Fortunately we show that simply assuming output regularity suffices in this case.

Assumption 2 (Regularity of outputs) *A function $h : \{0,1\}^m \rightarrow \{0,1\}^n$ is a ϵ* output regular function *if for any efficient machine A that gives a 1 bit output:*

$$|\Pr[A(x) = 1 \,|x \leftarrow h(U_m)] - \Pr[A(x) = 1 \,|x \leftarrow U_n]| \leq \epsilon$$

Here U_m and U_n denote the uniform distributions on $\{0,1\}^m$ and $\{0,1\}^n$, respectively.

We show that if the compression function is output regular (i.e. for a random input, the output is well distributed over the range) in addition to being collision-resistant, then it is secure against fixed points and thus a CRHF using the observation above.

Lemma 2.1 *The compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is $(\epsilon_{col} + \epsilon_{reg} + 2^{-n})$-secure against fixed points if it satisfies the following properties:*

- *h is ϵ_{col} collision resistant.*
- *h is an ϵ_{reg} output regular function.*

Proof: To the contrary, say there is an efficient attacker that finds a fixed point $x_1 \ldots x_i$ with non-negligible probability ϵ, then we can show that it either breaks the collision resistance or the output regularity assumption for the compression function. In order to show this, choose the initialization vector IV as $IV \leftarrow h(x)$ (for $x \leftarrow U_\kappa \times U_n$), instead of $IV \leftarrow U_n$. If the success probability of A changes by a non-negligible amount then we can break the output regularity assumption. Thus, $\epsilon' \geq \epsilon_{reg} + \Pr[A\ succeeds\ in\ new\ game]$.

To estimate the success probability of the attacker A in the new game, say it finds a sequence of κ-bit blocks $x_1 \ldots x_i$ such that $h(x_i, h(\ldots, h(x_1, IV)\ldots)) = IV$ with probability ϵ'. Let $y = (x_i, h(\ldots, h(x_1, IV)\ldots))$. Then it is the case that $h(x) = h(y)$ (where x was used to select the IV). Thus, we can deduce that,

$$\begin{aligned}
\epsilon' &= \Pr[(A\ succeeds) \wedge (x = y)] + \Pr[(A\ succeeds) \wedge (x \neq y)] \\
\Rightarrow \epsilon' &\leq \Pr[(x = y)] + \epsilon_{col} \\
\Rightarrow \epsilon' &\leq \epsilon_{col} + \sum_{IV \in \{0,1\}^n} \frac{\#\{x\ s.t.\ h(x) = IV\}}{2^{n+\kappa}} \cdot \frac{1}{\#\{x\ s.t.\ h(x) = IV\}} \\
&\leq \epsilon_{col} + 2^{-n}
\end{aligned}$$

Thus we get that the maximum success probability of an efficient fixed-point finding attacker is $\epsilon_{reg} + \epsilon_{col} + 2^{-n}$. □

Corollary 2.2 *The plain MD construction H using a compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a $(\epsilon_{reg} + \epsilon_{col} + 2^{-n})$ collision resistant hash function if h satisfies the following properties:*

- *h is ϵ_{col} collision resistant.*
- *h is an ϵ_{reg} output regular function.*

ENCODE-THEN-MD CONSTRUCTION. It makes sense to only consider deterministic input coding schemes, since the resulting construction must behave like a function. We analyze two of the most popular such coding schemes, i.e. *prefix-free encoding* and *suffix-free encoding*.

We first note that using a prefix-free encoding on the input does not enable us to get rid of any security properties in lemma 2.2. Hence we can essentially restate the same result for the prefix-free MD construction H_{pre} as well. On the other hand, if we use a *suffix-free encoding* (such as Merkle-Damgård strengthening) then the resulting suffix-free MD construction H_{suf} can be shown to be collision resistant by simply assuming the collision-resistance of the compression function h [10, 18].

MD-THEN-CHOP CONSTRUCTION Note that simply assuming collision resistance of the compression function is not useful for this construction, since we truncate s bits of the output. For instance, consider the case when h is collision resistant on these s bits, and is the constant function for all other bits (noted by Kelsey [15]). However, in our setting this only means that we need to make a stronger assumption on the compression function h. In particular, we will instead assume that h is collision resistant even if we remove these s bits from its output.

Lemma 2.3 *The MD-then-chop construction H_{chop_s}, using a compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$, is a $(\epsilon_{reg} + \epsilon'_{col} + 2^{n-s})$ collision resistant hash function if the following holds:*

- *The function $h' : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^{n-s}$ defined as $h'(x,y) = h(x,y)|_{n-s}$ (i.e. chopping the last s bits from the output of h) is a ϵ'_{col} collision resistant function.*
- *h is a ϵ_{reg} output regular function.*

The proof of this lemma is essentially the same as for corollary 2.2.

NMAC/HMAC CONSTRUCTION We note that using the NMAC construction H_{nmac} does not help in improving upon the collision resistance of the plain MD construction H. This is essentially because any collision in the first application of the plain MD construction of H_{nmac} (using initialization vector IV_1) essentially implies a collision for the entire construction. Hence, at best, we can restate lemma 2.2 for this construction as well.

Since the HMAC construction H_{hmac} is simply a black-box instantiation of the NMAC construction, this does not help in improving collision resistance. However, we note that it has the best exact security if $\alpha_1 \neq \perp$.

2.2 Pseudorandomness

An issue in the pseudorandomness analysis of the MD modes of operation is the location of the PRF key. As discussed above, we need to specify the location of the key such that the resulting construction is still a black-box variant of plain MD. For our analysis, we will assume the key length to be the length of a single block (i.e. κ bits for the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$), and we will denote the key as K. We will analyze two approaches for keying each MD mode of operation:

1. *Prepend the key to input:* The PRF construction H outputs $H(K \parallel X)$ on input X.
2. *Append the key to input:* The PRF construction H outputs $H(X \parallel K)$ on input X.

Moreover, we will need two versions of pseudorandomness definitions for the compression function, one where the key occupies the last n bits and other where it occupies the first κ bits. We get the following two assumptions on the compression function in this manner.

- *Standard PRF (sPRF) security:* Here we require that for a uniformly chosen $K \in \{0,1\}^n$, the function $h(\cdot, K)$ must be indistinguishable from a truly random function.
- *Dual PRF (dPRF) security:* Here we require that for a uniformly chosen $K \in \{0,1\}^\kappa$, the function $h(K, \cdot)$ must be indistinguishable from a truly random function.

Depending on the maximum distinguishing advantage ϵ of an efficient attacker in each case, we call the compression function h ϵ-sPRF or ϵ-dPRF.

PLAIN MD CONSTRUCTION. In this case if we prepend the PRF key to the hash function input, then the resulting construction is not a PRF. This is because an attacker can use the *extension attack* to find $H(K \parallel X \parallel Y)$ by simply knowing the output $H(K \parallel X)$ and computing the compression function on the remaining blocks itself (where it does not need to know the key K). On the other hand, if we append the PRF key to the input, then we can show that if the plain MD construction using h is collision-resistant and satisfies the dual PRF security, then the plain MD construction $\mathsf{H}(\cdot \parallel K)$ is a variable-length input PRF.

Lemma 2.4 *The plain MD construction* H *is a* $\mathcal{O}(\ell \cdot (\epsilon_{col} + \epsilon_{reg}) + \epsilon_{dprf})$ *PRF* [4] *(with PRF key appended to the function input) if the following conditions hold:*

- h *is* ϵ_{col} *collision resistant.*
- h *is a* ϵ_{reg} *output regular function.*
- h *is a* ϵ_{dprf} *dual pseudorandom function.*

The proof of this lemma is rather straightforward. Here, output regularity and collision resistance of the compression function together imply the collision resistance of the plain MD construction. Thus, in the last round, the n-bit chaining variable is different for two different inputs. Hence a distinguisher for the plain MD construction can be used directly by the dual-PRF distinguisher for the compression function.

ENCODE-THEN-MD CONSTRUCTION. Once again, we will discuss two deterministic coding schemes here, *prefix-free encoding* and *suffix-free encoding*. Let us first analyze the suffix-free MD construction H_{suf}. If we prepend the

[4] ℓ denotes the maximum number of κ-bit blocks in a hash function input, throughout this paper

key to the (encoded) input, the resulting construction is still insecure since the *extension attack* works in this case as well. On the other hand, if we append the key to the (encoded) input then the resulting construction is a PRF if the suffix-free MD construction H_{suf} using the compression function h is a dual PRF and collision resistant (for which we only need collision resistance of h in this case).

For the prefix-free MD construction H_{pre}, if we append the key to the (encoded) input then we get no advantage as compared to the plain MD construction and we can only restate lemma 2.4 in this case. On the other hand, if we prepend the PRF key to the (encoded) input then the resulting construction is not vulnerable to the *extension attack* in this case. Indeed, it was shown by Bellare et al. in [3] that the prefix-free MD construction with the PRF key in the IV is a PRF only assuming that the compression function h satisfies the standard PRF security. However, since we will need to prepend the key to the input (in order to preserve the black-box property of the construction), we will need to impose an extra condition on the compression function. In particular, we require that the function defined as $h(U_n, \cdot)$ is an output regular function. That is, if the first n bits of the compression function h are chosen at random then the resulting function is output regular with high probability.

Lemma 2.5 *The prefix-free MD construction H_{pre} is a $\mathcal{O}(\epsilon'_{reg} + \ell \cdot \epsilon_{sprf})$ secure PRF (with PRF key prepended to the input) if the following conditions hold:*

- *h is a ϵ_{sprf} sPRF.*
- *$h(U_n, \cdot)$ is a ϵ'_{reg} output regular function.*

The proof of this lemma is similar to the result of [3].

MD-THEN-CHOP CONSTRUCTION. If the PRF key is appended to the input to the MD-then-Chop construction H_{chop_s}, then a slight variant of lemma 2.4 can be stated for this construction as well. Indeed, all we need is to specify the dual PRF and collision-resistance properties for the compression function with chopped output.

On the other hand, if we prepend the PRF key to the input to H_{chop_s}, then the extension attack does not seem to go through as in the case of plain MD construction. This is because the attacker does not learn the chopped s bits of the chaining variable by observing the output of H_{chop_s} for the prefix of an input. Indeed, this construction can be proven to be an arbitrary-length input PRF by making a slightly non-standard assumption on the compression function. In particular, we require the compression function to satisfy the following *resilient sPRF* assumption:

Assumption 3 ((s, ϵ)-resilient sPRF) *The function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is a (s, ϵ)-resilient sPRF if it is a ϵ-secure sPRF even if the attacker learns s bits of the n bit key.*

Lemma 2.6 *The MD-then-Chop construction H_{chop_s} is a $\mathcal{O}(\epsilon'_{reg} + \ell \cdot \epsilon'_{sprf})$ secure PRF (with PRF key prepended to the input) if the following conditions hold:*

- *h is a (s, ϵ'_{sprf})-resilient sPRF.*
- *$h(U_n, \cdot)$ is a ϵ'_{reg} output regular function.*

NMAC/HMAC CONSTRUCTION. The NMAC and HMAC constructions were shown to be secure arbitrary-length input PRFs by Bellare [2]. In [2], it is shown that the HMAC construction with $\alpha_1 = \alpha_2 = \perp$ (i.e. with the same IV for both invocations of the plain MD construction) is a secure arbitrary-length input PRF if the underlying compression function satisfies both the standard and dual PRF security definitions. This is done by simply prepending a different κ-bit key to each invocation of the plain MD construction [5].

Lemma 2.7 *The NMAC (resp. HMAC) construction H_{nmac} (resp. H_{hmac}) is a $\mathcal{O}(q^2\ell \cdot \epsilon_{sprf} + \epsilon_{dprf})$ PRF (with a different κ-bit key prepended to the input in each call to the MD construction) for any IV_1 and IV_2 (resp. α_1 and α_2) if the following conditions hold:*

- *h is a ϵ_{sprf}-secure sPRF.*
- *h is a ϵ_{dprf}-secure dPRF.*

[5] if the same key is prepended in both invocations, then the construction is secure under a slightly stronger assumption, called security against *related-key attacks* in [3, 2]. We ignore this setting here

2.3 Target Collision Resistance

Target collision resistance (TCR) is a strictly weaker property than collision resistance. However, for some purposes, TCR hash functions (also called UOWHFs) suffice instead of CRHFs. For instance, it is possible to come up with a signature scheme on arbitrary length messages using one that works only for fixed-length messages by using TCR hash functions. For this reason, this primitive has attracted even greater interest since the discovery of better attacks against the collision resistance of existing hash functions. We will call a construction ϵ-secure TCR function if the maximum advantage of an efficient attacker in the TCR attack game is ϵ[6]

Here, simply assuming TCR security of the compression function will not suffice. This is because the output of a TCR function need not be random, so that each subsequent application of the compression function will require separate key space (and this dedicated-key setting violates our requirements from the mode of operation). Instead, we will assume that the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is an unkeyed function that satisfies second preimage resistance type properties.

PLAIN MD CONSTRUCTION. In order to discuss the TCR security of the plain MD construction, we need to first discuss appropriate keying mechanisms for this construction. As we briefly mentioned above, Shoup [24] described an efficient *masking-based construction* based on the plain MD construction. However, this construction modifies the chaining variable which violates our properties of black-box modes of operation. Unfortunately, we do not know of any black-box ways of keying the plain MD construction such that it can be shown to be a TCR hash function only assuming the compression function to be a SPR function.

Halevi and Krawczyk [13] suggested an alternate way of keying the plain MD construction that satisfies all the properties of a black-box mode of operation. The construction H_K proposed in [13] uses a κ-bit key K and XORs the key with each message block in the plain MD construction, i.e. $\mathsf{H}_K(x_1 \parallel \ldots \parallel x_\ell) \stackrel{def}{=} h(K \oplus x_\ell, h(\ldots, h(K \oplus x_1, IV) \ldots))$.

However, in order to prove TCR security of this construction one needs to make a slightly non-standard "SPR type" assumption on the compression function, called the *evaluated SPR assumption* (e-SPR) [13].

Assumption 4 (evaluated SPR) *Consider a function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ and let H_K be the plain MD based construction using h (described above). The function h is ϵ* evaluated second preimage resistant *if any efficient machine A wins in the following game with probability at most ϵ (over the random choice of IV and the coins of A).*

1. *A chooses a sequence of κ-bit blocks $\Delta_1, \ldots, \Delta_i$.*
2. *The challenger chooses a random key K and sets $c = \mathsf{H}_K(\Delta_1 \oplus K, \ldots, \Delta_{i-1} \oplus K)$ and $m = \Delta_i \oplus K$.*
3. *A wins if it can find c' and m' such that $h(m', c') = h(m, c)$.*

Halevi and Krawczyk [13] show that if the compression function h is an e-SPR function, then the construction H_K described above is a secure TCR hash function. However, in their proof they require that the inputs provided to H_K must be suffix-free. Indeed, this is required for their reduction to go through. However, we note that even for the plain MD construction (with possibly "non-suffix-free" inputs), one can make an additional assumption on the compression function to enable us to apply the proof technique of [13].

Lemma 2.8 *The construction H_K is an $\mathcal{O}(\ell \cdot (\epsilon_{fix} + \epsilon_{espr}))$-secure TCR function if the following holds:*

- *h is an ϵ_{espr}-secure e-SPR function.*
- *h is ϵ_{fix}-secure against fixed points.*

ENCODE-THEN-MD CONSTRUCTION. If we apply a suffix-free encoding to the input before using the construction, then the resulting mode of operation $\mathsf{H}_{suf,K}$ is a TCR hash function based only on the assumption that h is an e-SPR function [13]. On the other hand, using a prefix-free encoding does not help in improving the security of the plain MD construction and we need all conditions of lemma 2.8 to prove the TCR security of the resulting construction.

[6]Recall that in this game, the attacker first chooses a function input, followed by the challenger choosing the key and finally the attacker winning if it finds a second colliding input for the chosen key.

MD-THEN-CHOP CONSTRUCTION. For the MD-then-Chop construction, we need to make a slightly stronger assumption on the compression function to prove the TCR security of the resulting construction. In particular, we need to assume that the compression function h is e-SPR even if we chop a non-negligible number of its output bits. If we replace the second condition in lemma 2.8 wit this stronger condition, then it holds for the MD-then-Chop construction as well.

NMAC/HMAC CONSTRUCTION. Using the NMAC or HMAC construction does not lead to improved TCR security of the resulting construction. Again this is because if the attacker finds a collision in the first invocation of the plain MD construction then it implies a collision for both NMAC and HMAC construction.

2.4 One-Wayness

One way functions are also often referred to as preimage resistant functions. A construction is ϵ-secure OWF if no efficient attacker can find the input corresponding to the output of the function (on a random input) with probability more than ϵ. This security property is even weaker than second preimage resistance.

PLAIN MD CONSTRUCTION. In this case, we will need to assume that the compression function h is a one way function. Moreover, we will also require that h is output regular, so that its output is uniformly distributed for a random input. This is essentially because we need the input to a one-way function to be random in order to use the one-wayness property.

Lemma 2.9 *The plain MD construction* H *is* $\mathcal{O}(\ell \cdot \epsilon_{reg} + \epsilon_{owf})$*-secure OWF if the following conditions hold:*

- *h is an ϵ_{reg} output regular function.*
- *h is a ϵ_{owf}-secure one-way function.*

The proof of this lemma is based on the fact that an attacker cannot tell the difference between the output of H on a random input or the compression function h on a random input, if h is output regular. Thus the one-wayness attacker for h can use the one for H directly.

ENCODE-THEN-MD CONSTRUCTION. If we use an arbitrary suffix-free encoding with the MD construction, then we cannot say much about one-wayness of the construction since the input distribution could be arbitrary. However, if we apply *Merkle-Damgård strengthening* to the input, then we can show that the resulting construction is a one-way function under sufficient assumptions. The proof of this fact is non-trivial though. In particular, we need to make an additional assumption about the compression function.

Assumption 5 ((p, ϵ) output consistent) *The function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is (p, ϵ) output consistent if for any κ-bit block x and uniformly distributed $y \in \{0,1\}^n$, with probability at least $(1 - \epsilon)$ the number of $y' \in \{0,1\}^n$ such that $h(x, y) = h(x, y')$ is at most p.*

Note that this property certainly holds for a random compression function (and, thus, holds for most compression functions). By making this additional assumption from the compression function, we can derive the following result.

Lemma 2.10 *The suffix-free MD construction H_{suf} that uses MD strengthening for suffix-freeness is $(p_{cons} \cdot (\ell \cdot \epsilon_{reg} + \epsilon_{owf}) + \epsilon_{cons})$-secure one-way function, where ℓ is the maximum length of an inverted input provided by the OWF attacker, if the following conditions hold:*

- *h is an ϵ_{reg} output regular function.*
- *h is a ϵ_{owf}-secure one-way function.*
- *h is a $(p_{cons}, \epsilon_{cons})$ output consistent.*

Proof: The proof for this lemma is essentially based on the proof of lemma 2.9. We construct an one-wayness attacker A' for the compression function using the attacker A that has advantage ϵ in inverting H_{suf} with MD strengthening. A' gets its challenge output y and chooses a uniformly random $i \in \{1, \ldots, \ell\}^n$. It then gives $z = h(\langle i \rangle, y)$ as a challenge to A.

Now A' succeeds only if the inverse z outputted by A is i-bit long. If so, then A' can proceed similar to the case on the plain MD construction in lemma 2.9 if the chaining variable for z in the last round, with $\langle i \rangle$ in the message block, is the challenge y. However, from our assumptions, with probability at most ϵ_{cons} there are more than p_{cons} n-bit strings y' such that $h(\langle i \rangle, y') = h(\langle i \rangle, y)$. Thus, we get that the success probability of A is at most $(p_{cons} \cdot (\ell \cdot \epsilon_{reg} + \epsilon_{owf}) + \epsilon_{cons})$. □

As for prefix-free encoding, once again we cannot say anything general (for the same reason as above), but when prepending the message length we are essentially back to the setting of plain MD discussed above, except we need to assume that the output of the compression function on a random IV and a fixed message block is random. In particular, we note that encoding the input in any way does not help as far as one-wayness of the construction is concerned. In fact, we only need more assumptions to prove this property, as compared to the plain MD construction.

MD-THEN-CHOP CONSTRUCTION. In order to prove the one-wayness of the MD-then-Chop construction, we need to make a stronger assumption on the compression function h. In particular, we assume that h is one-way with s bits of the output chopped. Let the one-way security of the function h with truncated output be ϵ'_{owf}. Then we can show that H_{chop_s} is a $\mathcal{O}(\ell \cdot \epsilon_{reg} + \epsilon'_{owf})$-secure one-way function (similar to lemma 2.9)

NMAC/HMAC CONSTRUCTION. The NMAC construction is a one-way function under the same conditions on the underlying compression function h as required in lemma 2.9. However, we require that random and independent initialization vectors IV_1 and IV_2 are used in the NMAC construction. However, it turns out that translating these results to the setting of the HMAC construction is not straightforward.

3 Implications for Hash Functions in Practice

We will now translate our results into suggestions for usage of actual "cascade construction based" hash functions, such as functions from the SHA family. As we mentioned earlier, we have tried to find the minimal assumptions needed to make each of the four modes of operation secure (for each of the security properties). Thus, we have left part of the "decision making" for the practitioner who uses our results. In particular, the practitioner must consider the following questions:

1. What one needs to assume about the hash function in order for the cryptosystem (that the hash function is being used for) to be provably secure?
2. What level of trust the practitioner is willing to place in the underlying compression function?

The answer to the first question will help in deciding the security property to look for in the hash function mode of operation. The answer to the second question may not be as straightforward since the design of the compression functions is quite complex and mostly based on heuristic. In this case, the practitioner needs to weigh all the properties (s)he desires from the cryptosystem, in terms of efficiency, security etc. Thus, while some may be willing to make a slightly stronger assumption on the compression function to have a more efficient implementation, others may be willing to sacrifice some efficiency for better security. Now we will give some basic recommendations for actual hash functions with respect to the various security properties.

COLLISION RESISTANCE. Each of the SHA functions are essentially based on the suffix-free MD construction (using MD strengthening). Hence, collision resistance for each of these hash functions is asymptotically same as finding collisions on the compression function. It does not make much sense to use the "truncated" versions, SHA-224 and SHA-384, since this only sacrifices the collision resistance of the original "untruncated" version (i.e. SHA-256 and SHA-512, respectively). Using the NMAC/HMAC construction does not help in this case.

PSEUDORANDOMNESS. We note that using the full SHA-256 or SHA-512 hash functions makes more sense for pseudorandomness than using the chopped versions (SHA-228 or SHA-384), which only have worse security. If any of the SHA functions are used, as it is, for pseudorandomness, then we recommend appending the PRF key to the input instead of prepending it. However, we recommend using these functions in conjunction with a prefix-free encoding (such as prepending input length to the input) in which case the PRF key should be *prepended* to the input. Another option would be to compose two calls to SHA-1, with independent keys prepended in each call, to get security based on the sPRF and dPRF security of the compression function.

RANDOM ORACLE. Note that none of the SHA functions should be used, as it is, if the security of the cryptosystem requires the *random oracle assumption* for the hash function. This is because the plain MD construction (even with

MD strengthening) is vulnerable to simple attacks in the indifferentiability scenario. One may think that both SHA-224 and SHA-384 that correspond to "chop" versions of the functions SHA-256 and SHA-512 would be secure (since the MD-then-Chop construction is secure). However, note that only 32 bits are chopped in the case of SHA-224, which does not give sufficient security for almost all applications. Hence, only SHA-384 (that chops 128 bits) may be suitable to be used directly to instantiate the random oracle.

We recommend using the HMAC construction involving two black-box calls to the SHA function (while prepending different α_1 and α_2 in each cal) for this purpose. Using any of these hash functions in conjunction with a prefix-free encoding will also work for this purpose.

MESSAGE AUTHENTICATION. If the SHA functions are used as MACs directly, then the MAC key should be appended to the input. In this case, security depends on both the MAC security and collision resistance of the compression function. Using the HMAC construction does not help in improving the security either. Moreover, when the "chopped" functions SHA-224 or SHA-384 are used as MACs, then their security is only worse than the unchopped versions (SHA-256 and SHA-512).

If one is willing to assume pseudorandomness of the compression function, then the techniques mentioned above for pseudorandomness can be used as well. Another approach would be to assume the *dedicated-key setting*, by inserting the MAC key in each application of the compression function (at the cost of some input bandwidth) and then one could use one of the techniques suggested in [1, 20].

TARGET COLLISION RESISTANCE OR UOWHFS. We recommend using the technique suggested by Halevi and Krawczyk [13] if the SHA functions are used as UOWHFs. In this case, one XORs the UOWHF key to each block of the input. Since MD strengthening is already used in all these functions, the UOWHF security of this construction is based only on the eSPR [13] (see above) of the compression function.

SECOND PREIMAGE RESISTANCE. It makes sense to use the SHA hash functions directly for the purpose of *second preimage resistance* without using any additional techniques, since they do not lead to improved security (note that these functions already incorporate MD strengthening).

RANDOMNESS EXTRACTION. All the positive results for *randomness extraction* have reasonable interpretation in practice, only if we are willing to assume that the SHA compression function is close to being a family of random functions. Even though it is theoretically impossible, since the SHA compression function has a short description, it might still be a more reasonable assumption than assuming the compression function to be a FIL-RO.

Under this assumption, we can deduce that the SHA functions are good randomness extractors for input distributions with high min entropy overall and in the last block. On the other hand, as we saw above, it might be a good idea to use chopped function SHA-384 for this purpose to get better extraction properties (SHA-224 does not have sufficient number of chopped bits to give useful advantage). Using the HMAC construction does not help in improving the extraction properties.

ONE-WAYNESS. In the case of "one-wayness", the security of the chopped functions, SHA-224 and SHA-384, seems to rely on stronger assumptions than the security of the corresponding "unchopped" versions (SHA-256 and SHA-384). This is because the one-way security increases with the number of output bits. On the other hand, it might be the case that SHA-224 still has higher security than SHA-1, which seems intuitive given the bigger IV of SHA-224. Moreover, message encoding or HMAC construction only negatively affects the one-wayness.

4 Conclusions

In this work we showed how to efficiently use existing hash functions based on the MD mode (such as the functions in the SHA family) to build cryptographic hash functions satisfying various security properties such as collision-resistance, pseudorandomness, indifferentiability from random oracle, message authentication, target collision-resistance, second preimage-resistance, randomness extraction and one-wayness. Our constructions are black-box, support variable-length inputs and provide the same efficiency as the plain MD construction, under the minimal assumptions on the underlying compression function.

References

[1] J. H. An, M. Bellare, *Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions*, CRYPTO 1999,pages 252-269.

[2] M. Bellare, *New Proofs for NMAC and HMAC: Security without Collision-Resistance*, Advances in Cryptology - Crypto 2006 Proceedings, Springer-Verlag, 2006.

[3] M. Bellare, R. Canetti, and H. Krawczyk, *Pseudorandom Functions Re-visited: The Cascade Construction and Its Concrete Security*, In Proc. 37th FOCS, pages 514-523. IEEE, 1996.

[4] M. Bellare, R. Canetti, and H. Krawczyk, *Keying hash functions for message authentication*, Advances in Cryptology - Crypto 96 Proceedings, LNCS Vol. 1109, Springer-Verlag, 1996.

[5] M. Bellare and P. Rogaway, *Collision-Resistant Hashing: Towards Making UOWHFs Practical*, In *Crypto '97*, LNCS Vol. 1294.

[6] M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols.* Proceedings of the First Annual Conference on Computer and Communications Security, ACM, 1993.

[7] M. Bellare and T. Ristenpart, *Multi-Property-Preserving Hash Domain Extension and the EMD Transform*, in ASIACRYPT 2006.

[8] M. Bellare and T. Ristenpart, *Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms*, in ICALP 2007.

[9] J.-S. Coron, Y. Dodis, C. Malinaud and P. Puniya, *Merkle-Damgård Revisited: How to Construct a Hash Function*, Advances in Cryptology, Crypto 2005 Proceedings: 430-448, Springer-Verlag, 2006.

[10] I. Damgård, *A Design Principle for Hash Functions*, In Crypto '89, pages 416-427, 1989. LNCS No. 435.

[11] Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin, *Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes*, Advances in Cryptology - CRYPTO, August 2004.

[12] FIPS 180-1, *Secure hash standard*, Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, April 17 1995 (supersedes FIPS PUB 180).

[13] S. Halevi and H. Krawczyk, *Strengthening Digital Signatures Via Randomized Hashing*, in Advances in Cryptology - CRYPTO 2006, pp. 41-59.

[14] Antoine Joux, *Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions*, Advances in Cryptology - CRYPTO 2004, 306-316.

[15] J. Kelsey, in CRYPTO 2005 Rump Session.

[16] J. Kelsey and T. Kohno, *Herding Hash Functions and the Nostradamus Attack*, Advances in Cryptology - EUROCRYPT 2006, 183-200.

[17] RFC 1321, *The MD5 message-digest algorithm*, Internet Request for Comments 1321, R.L. Rivest, April 1992.

[18] R. Merkle, *One way hash functions and DES*, Advances in Cryptology, Proc. Crypto'89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428-446.

[19] U. Maurer, R. Renner, and C. Holenstein, *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*, Theory of Cryptography - TCC 2004, Lecture Notes in Computer Science, Springer-Verlag, vol. 2951, pp. 21-39, Feb 2004.

[20] U. M. Maurer, J. Sjödin, *Single-Key AIL-MACs from Any FIL-MAC*, in ICALP 2005, pp. 472-484.

[21] National Institute of Standards and Technology, *NIST's Plan for New Cryptographic Hash Functions*, http://www.csrc.nist.gov/pki/HashWorkshop/index.html.

[22] Moni Naor and Moti Yung, *Universal One-Way Hash Functions and their Cryptographic Applications*, STOC 1989: 33-43.

[23] D. R. Simon, *Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?*, in Advances in Cryptology - EUROCRYPT 1998, pp. 334-345.

[24] Victor Shoup, *A Composition Theorem for Universal One-Way Hash Functions*, in Advances in Cryptology - EUROCRYPT 2000, pp. 445-452.

[25] X. Wang, H. Yu, Y. L. Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology - CRYPTO 2005, 1-16.

[26] X. Wang, Y. L. Yin, H. Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology - CRYPTO 2005: 17-36.

A Preliminaries

In this paper, we will be interested more in the qualitative aspects of the security of iterative hash functions rather than focusing on the exact security in each case. For this purpose, we will give here slightly "less formal" and asymptotic definitions for each of these security notions related to hash functions.

A.1 Collision Resistance

A collision resistant function ensemble H_λ is defined for a sequence of sets $\left\{\{0,1\}^{m(\lambda)}, \{0,1\}^{n(\lambda)}\right\}_{\lambda\in\mathbb{N}}$, where m and n denote the input and output length of H_λ, respectively. Such a function ensemble consists of a pair of PPT machines $(Gen, Eval)$. However, we will give an asymptotic version of the definition of collision resistance here.

Definition 1 *ϵ-CR function family A function ensemble H_λ is a ϵ-collision resistant function family if for any probabilistic polynomial time machine A:*

$$\Pr\left[h_s(x_1) = h_s(x_2)\,\middle|\, s \leftarrow Gen(1^\lambda);\ (x_1, x_2) \leftarrow A(1^\lambda, s)\right] \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

A.2 Pseudorandomness

Apseudorandom function ensemble H_λ is defined for a sequence of sets $\left\{\{0,1\}^{m(\lambda)}, \{0,1\}^{n(\lambda)}\right\}_{\lambda\in\mathbb{N}}$. It consists of a pair of PPT machines $(Gen, Eval)$, the key generation and evaluation machines.

Definition 2 *ϵ-PRF family Let R_λ be the truly random function ensemble. A function ensemble H_λ is a ϵ-pseudorandom function family if for any PPT oracle machine A:*

$$\left|\Pr\left[A^{h_s}(1^\lambda) = 1 \,\middle|\, s \leftarrow Gen(1^\lambda)\right] - \Pr\left[A^f = 1 \,|\, f \leftarrow R_\lambda\right]\right| \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

A.3 Unpredictability and MACs

A message authentication code, MAC, is defined for a sequence of sets $\{\mathcal{M}_\lambda, \mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$. It consists of a triple (Gen, Tag, Ver) of PPT machines, denoting the key generation, tagging and tag verification algorithms.

Definition 3 (ϵ-secure MAC) *A MAC (Gen, Tag, Ver) is a ϵ-secure MAC if for any PPT oracle machine A that outputs a message/tag pair (m, t) such that it never queried the tagging oracle on the message m:*

$$\Pr\left[Ver_s(m,t) = accept \,\middle|\, \begin{array}{c} s \leftarrow Gen(1^\lambda); \\ (m,t) \leftarrow A^{Tag_s, Ver_s}(1^\lambda) \end{array}\right] \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

A.4 TCR and One-Wayness

Target collision resistance is a weaker notion of collision intractability that collision resistance. A target collision resistant function ensemble is also called a *Universal One-Way Hash Function* ensemble (or simply UOWHFs). A TCR function ensemble is defined for a sequence of sets $\left\{\{0,1\}^{m(\lambda)}, \{0,1\}^{n(\lambda)}\right\}_{\lambda\in\mathbb{N}}$, and consists of a pair of algorithms $(Gen, Eval)$. However, the TCR attacker is more restricted than the collision finding attacker above, since it chooses one of the colliding inputs without knowledge of the hash function key.

Definition 4 (ϵ-TCR function family) *A function ensemble H_λ is a ϵ-secure TCR function family if for any pair of PPT machines (A_1, A_2):*

$$\Pr\left[h_s(x_1) = h_s(x_2) \,\middle|\, \begin{array}{c} (x_1, \alpha) \leftarrow A_1(1^\lambda);\ s \leftarrow Gen(1^\lambda); \\ x_2 \leftarrow A_2(1^\lambda, \alpha, x_1, s) \end{array}\right] \leq \epsilon$$

Here ϵ is a function of the security parameter λ.

A notion related to TCR hash functions is that of *second preimage-resistant functions*. Unlike TCR hash functions this security notion is related to unkeyed hash functions $f : \{0,1\}^m \to \{0,1\}^n$ (where we can think of m as being the security parameter).

Definition 5 (ϵ-SPR function) *A function $f : \{0,1\}^m \to \{0,1\}^n$ is* ϵ-second preimage resistant *if for any PPT machine A:*

$$\Pr\left[f(x) = f(x') \,\middle|\, x \xleftarrow{\$} \{0,1\}^m;\ x' \leftarrow A(1^\lambda, x)\right] \leq \epsilon$$

Related to the notion of SPR functions, we can also define the notion of *preimage resistance* or *one-wayness*. This is a slightly weaker property than second preimage resistance.

Definition 6 *ϵ-secure one way function A function $f : \{0,1\}^m \to \{0,1\}^n$ is an* ϵ-secure one way function *if for any PPT machine A:*

$$\Pr\left[f(x) = y \middle| y \xleftarrow{\$} \{0,1\}^n;\ x \leftarrow A(1^\lambda, y)\right] \leq \epsilon$$

A.5 Randomness Extraction

A *randomness extractor* is a function that is used to extract uniformly random bits from inputs samples from an imperfect source of randomness. This has been an extremely useful primitive in cryptography, as well as theoretical computer science in general. We will give here brief definitions for this primitive.

We start by defining the notion of *min entropy*, which is a measure of the amount of randomness in a probability distribution. For instance consider a distribution $\mathcal{X}$ over $\{0,1\}^n$. The *min entropy* of the distribution $\mathcal{X}$, denoted as $H_\infty(\mathcal{X})$, is the minimum integer m such that $\Pr_\mathcal{X}(x) \leq 2^{-m}$ for all $x \in \{0,1\}^n$. Here $\Pr_\mathcal{X}(x)$ denotes the probability assigned to x by the distribution $\mathcal{X}$.

We will also need a way to quantify the distance between two probability distributions, $\mathcal{X}_1$ and $\mathcal{X}_2$, over a set S. The popular measure in this case is *statistical distance* between $\mathcal{X}_1$ and $\mathcal{X}_2$. The *statistical distance* between $\mathcal{X}_1$ and $\mathcal{X}_2$ is defined as $\mathbf{SD}(\mathcal{X}_1, \mathcal{X}_2) \stackrel{def}{=} \frac{1}{2}\sum\limits_{s \in S} |\Pr_{\mathcal{X}_1}(x) - \Pr_{\mathcal{X}_2}(x)|$. If two distributions have statistical distance ϵ between them, then they are called ϵ-close distributions.

A *randomness extractor* is a function $h : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ that takes a κ-bit uniformly random seed and a m-bit input, and outputs a n-bit output.

Definition 7 (**(k,ϵ) Extractor**) *A (k,ϵ) extractor is a function $f : \{0,1\}^\kappa \times \{0,1\}^m \to \{0,1\}^n$ such that for every distribution $\mathcal{X}$ on $\{0,1\}^\kappa$ with $H_\infty(\mathcal{X}) \geq k$, the distribution $f(\mathcal{X}, U_m)$ is ϵ-close to the uniform distribution on $\{0,1\}^n$, where U_m denotes the uniform distribution on $\{0,1\}^m$.*

B Random Oracle

Now we analyze each of the hash function modes of operation for indifferentiability from a random oracle (RO), under the assumption that the underlying compression function is a fixed-length input random function oracle (FIL-RO). We will call a function ϵ RO if the maximum advantage of a distinguisher is ϵ under the indifferentiability definition.

PLAIN MD CONSTRUCTION. The plain MD construction does not give an indifferentiable construction of RO from a FIL-RO. This is essentially because the plain MD construction is vulnerable to the extension attack (see [9]).

ENCODE-THEN-MD CONSTRUCTION. The suffix-free MD construction H_{suf} is not indifferentiable from a random oracle, since applying a suffix-free encoding to the input does not help in avoiding the *extension attack* [9]. However, if the input to the MD construction is guaranteed to be prefix-free, then it is no longer vulnerable to the extension attack. Indeed, it was shown by Coron et al [9] that the prefix-free MD construction H_{pre} is indifferentiable from a random oracle if the underlying compression function is a FIL-RO.

Lemma B.1 *The prefix-free Merkle-Damgård construction H_{pre} is a $\mathcal{O}((q\ell)^2/2^n)$ RO if the underlying compression function is a FIL-RO.*

MD-THEN-CHOP CONSTRUCTION. The Merkle-Damgård construction can be shown to be indifferentiable from RO if we chop a non-negligible (in particular, super-logarithmic) number of the output bits. However, the security of the MD-then-chop construction H_{chop_s} is worse than other MD based constructions [9] (in particular, it is a birthday bound on s bits instead of on n bits).

Lemma B.2 *The Merkle-Damgård then chop construction H_{chop_s} is a $\mathcal{O}((q\ell)^2/2^s)$ RO if the underlying compression function is a FIL-RO, and we chop s bits of the output.*

NMAC/HMAC CONSTRUCTION. It was shown in [9] that the HMAC construction with $\alpha_1 = 0^k$ and $\alpha_2 = \perp$ is indifferentiable from a random oracle. We note that α_1 can be any k-bit block such that $\alpha_1 \notin \{\perp, \alpha_2\}$, while α_2 can be any bit string in $\{\perp\} \cup \{0,1\}^k$. On the other hand, it is more straightforward to show that the version of NMAC construction H_{nmac} described here, is indifferentiable from a random oracle as well, with $IV_1 \neq IV_2$.

Lemma B.3 *The NMAC/HMAC constructions H_{nmac} (resp. H_{hmac}) with $IV_1 \neq IV_2$ (resp. $\alpha_1 \notin \{\perp, \alpha_2\}$) are $\mathcal{O}((q\ell)^2/2^n)$ ROs if the underlying compression function is a FIL-RO.*

C Message Authentication Code

We will refer to MACs that work for fixed-length messages as FIL-MACs and those that work for variable-length messages as VIL-MACs. We will analyze each of the modes of operation to see if they satisfy VIL-MAC security. We call a construction ϵ-secure VIL-MAC if the maximum advantage of an efficient attacker in producing a valid forgery is ϵ. Let us first note, that a pseudorandom function can be considered to be a MAC as well (PRF output serves as the message tag). Thus all the results for PRF security above hold for VIL-MAC security as well. We will try to find if these modes are VIL-MACs under weaker assumptions on the compression function than those needed for the case of PRFs.

However, if we assume the compression function to be simply a FIL-MAC, then we cannot use the output of one application of the compression function to key the construction. One solution to this problem would be to analyze the construction in the *dedicated-key setting*, where each call to the compression function has a separate key space. For current hash functions, one could assume that part of the message block space can be used to securely key the compression function. That is, for the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, the key occupies part of the first κ bits in the input. In this case, we can use the results of [1, 20] to get secure VIL-MAC constructions. However, as we discussed earlier, this violates the property that our modes of operation should be efficient in terms of input bandwidth. Thus, we will take a different approach here.

PLAIN MD CONSTRUCTION. If we prepend the MAC key $K \in \{0,1\}^\kappa$ to the input and apply the plain MD construction, then the resulting construction is vulnerable to the extension attack since the attacker can obtain the tag for a message by first getting a tag for the prefix. On the other hand, if the MAC key is appended to the input, then we find sufficient assumptions to show that H is a secure VIL-MAC. In particular, we will need the plain MD construction to be collision-resistant and the compression function to be a secure MAC when the MAC key occupies the first κ bits of its input.

Lemma C.1 *The plain MD construction H is a $\mathcal{O}(\ell\cdot(\epsilon_{reg}+\epsilon_{col})+\epsilon_{mac})$-secure VIL-MAC, when the key is appended to the input, if the following conditions hold:*

- *h is a ϵ_{col} collision resistant function.*
- *h is a ϵ_{reg} output regular function.*
- *h is a ϵ_{mac} secure MAC, when the first κ bits of its input is considered to be the key space.*

The proof idea here is similar to the corresponding PRF lemma 2.4.

ENCODE-THEN-MD CONSTRUCTION. If we use a suffix-free encoding and append the MAC key to the input, then we succeed in reducing the assumptions needed in lemma C.1 for collision resistance. Indeed, in this case, we can show that the suffix-free MD construction H_{suf} is $\mathcal{O}(\epsilon_{mac} + \ell \cdot \epsilon_{col})$ secure VIL-MAC if the compression function is

ϵ_{col} collision resistant and ϵ_{mac} secure FIL-MAC. On the other hand, if we prepend the MAC key to the input, then the resulting construction is insecure since it is still vulnerable to the extension attack.

If we use a prefix-free encoding, and prepend the MAC key to the input then the resulting construction is a secure VIL-MAC only if all the conditions stated in lemma 2.5 hold. On the other hand, the prefix-free MD construction H_{pre} with MAC key appended to the input essentially has the same security as the plain MD construction in lemma C.1.

MD-THEN-CHOP CONSTRUCTION. If we prepend the MAC key to the input to the MD-then-Chop construction H_{chop_s}, then the resulting construction can be shown to be a VIL-MAC only under the conditions from lemma 2.6. On the other hand, if we append the MAC key to the input then we can prove the VIL-MAC security of the resulting construction by making slightly stronger assumptions on the compression function as compared to lemma C.1.

Lemma C.2 *The MD-then-Chop construction H_{chop_s} is $\mathcal{O}(\ell \cdot (\epsilon_{reg} + \epsilon_{col}) + \epsilon'_{mac})$-secure VIL-MAC if the following conditions hold:*

- *h is a ϵ_{col} collision resistant function.*
- *h is a ϵ_{reg} output regular function.*
- *$h'(\cdot) = h(\cdot)|_{n-s}$ is a ϵ'_{mac} secure FIL-MAC.*

NMAC/HMAC CONSTRUCTION. In this case, if we prepend the MAC key to the input, then we need the same conditions as lemma 2.7 in order to prove VIL-MAC security as well. On the other hand, if we append the MAC key to the input, then both NMAC and HMAC constructions can only be proven secure using the same conditions as in the case of plain MD construction (lemma C.1).

D Second Preimage Resistance

In this section, we will analyze each of the modes of operation for the minimal assumptions on the compression function needed in order to prove the SPR security of the construction. A construction will be called ϵ-secure SPR function if any efficient attacker has success probability at most ϵ in the SPR attack game.

Unfortunately, to the best of our knowledge, there is no black-box mode of operation that is property preserving for second preimage resistance. Hence we will need to make a slightly stronger assumption on the compression function h.

Assumption 6 (computed SPR (cSPR) [13]) *A function $h : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a ϵ-*secure cSPR function *if any efficient machine A has success probability at most ϵ in the following game:*

1. *The challenger randomly selects a sequence of κ-bit blocks $x_1, \ldots, x_\ell$, sets $c = \mathsf{H}(x_1 \parallel \ldots \parallel x_{i-1})$ and $x = x_i$. Here H is the plain MD construction using the compression function h and random IV. The challenger sends $x_1, \ldots, x_i$ to A.*
2. *A wins if it finds $(x', c') \in \{0,1\}^\kappa \times \{0,1\}^n$ such that $h(x', c') = h(x, c)$.*

Note that this assumption is quite similar to the eSPR assumption that we needed for the TCR hash function case.

PLAIN MD CONSTRUCTION. Here we will assume that the compression function h is a cSPR function. However, in order to prove SPR security of the plain MD construction, we will also need to assume that the attacker cannot find fixed points starting from a random IV.

Lemma D.1 *The plain MD construction* H *using the compression function* h *is a* $\mathcal{O}(\ell \cdot (\epsilon_{cspr} + \epsilon_{fix}))$*-secure SPR function if the following conditions hold:*

- h *is a* ϵ_{cspr}*-secure cSPR function.*
- h *is* ϵ_{fix} *secure against fixed points.*

ENCODE-THEN-MD CONSTRUCTION. If we use a suffix-free encoding on the input, then the resulting construction H_{suf} can be proven to be a SPR function solely on the assumption that the compression function h is a cSPR function. On the other hand, the prefix-free MD construction does not help in gaining any improvement in SPR security over the plain MD construction.

Lemma D.2 ([13]) *The suffix-free MD construction* H_{suf} *is a* $\mathcal{O}(\ell \cdot \epsilon_{cspr})$*-secure SPR function if the compression function* h *is a* ϵ_{cspr}*-secure cSPR function.*

MD-THEN-CHOP CONSTRUCTION. As in the case of collision resistance and TCR functions, we will need to impose a slightly stronger assumption on the compression function in order to prove the SPR security of the MD-then-Chop construction H_{chop_s}. In particular, we will need to assume that the function $h'(\cdot) = h(\cdot)|_{n-s}$ is ϵ'_{cspr}-secure SPR function. This assumption along with the second condition from lemma D.1 suffices to show that the MD-then-Chop construction is a $\mathcal{O}(\ell \cdot (\epsilon_{fix} + \epsilon'_{cspr}))$-secure SPR function.

NMAC/HMAC CONSTRUCTION. The NMAC/HMAC construction do not give any better SPR security as compared to the plain MD construction. This is because a collision in the first invocation of the MD construction implies a collision for both the NMAC and HMAC constructions.

E Randomness Extraction

The idea of using the MD construction as a randomness extractor was discussed by Dodis et al in [11]. They showed that for getting any useful randomness extraction properties from the MD construction, one needs to make a really strong assumption on the compression function h. In particular, they assume that the compression function $h : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ is an *ideal randomness extractor*, which is the same as assuming it to be a *family of random functions*[7]. That is, the function $h(\cdot, x)$ is a random function from κ to n bits when x is uniformly distributed. We debate such a compression function h as a family of random functions $\{h_r\}$ for $r \in \{0,1\}^n$.

Let us start by explaining why one needs to make such a strong assumption on h. If we assume h to be a regular extractor, then the distribution of the output of $h(\cdot, x)$ for random x has a non-zero statistical distance from the uniform distribution on $\{0,1\}^n$. If this output is used a seed for the next application of the compression function then one has no guarantee of extraction, since the seed us no longer independent of the κ-bit input block. Actually, Dodis et al [11] do give a positive result for the MD construction simply under the assumption that h is an *almost-universal family of functions* [8]. However, for this result they require that every input block for the MD construction must have some amount of *conditional min-entropy*[9] (see [11] for more details). However, all the results here are based on the assumption that h is an ideal randomness extractor.

PLAIN MD CONSTRUCTION. In this case, one can show that for a restricted class of inputs (from certain high min entropy distributions), the output of the plain MD construction, using an *ideal randomness extractor* h, is close to uniform. The input distribution should be such that it has high overall min entropy as well as high conditional min-entropy in the last input block.

[7] Note that this is a weaker assumption than assuming h to be a FIL-RO. In particular, it is a (very inefficiently) realizable assumption.

[8] i.e., for the function $h(\cdot, x)$, where x uniformly distributed on $\{0,1\}^n$, for any two distinct inputs y and z, the probability that $h(z, x) = h(y, x)$ is negligibly close to the corresponding probability for a random function

[9] This is the min entropy of an input block conditioned on all the other input blocks.

Lemma E.1 ([11]) *Let $\{\mathsf{H}_r\}$ be the plain MD construction defined over a family of random functions $\{h_r\}$, where the seed r is essentially the random IV in the plain MD construction. Let $\mathcal{X}$ be the distribution of the inputs to H (over bit strings with at most ℓ κ-bit blocks) and let $\mathcal{X}_\ell$ be the distribution induced by $\mathcal{X}$ on the last block of the input. If $H_\infty(\mathcal{X}) > n + 2\log\left(\frac{1}{\epsilon}\right)$ and $H_\infty(\mathcal{X}_\ell) > \log \ell + 2\log\left(\frac{1}{\epsilon}\right)$, then $\mathbf{SD}(\mathsf{H}_{\mathbf{IV}}(\mathcal{X}), \mathbf{U_n}) = \mathcal{O}(\epsilon)$ where U_n is the uniform distribution on n-bit strings.*

ENCODE-THEN-MD CONSTRUCTION. Note that if one applies a suffix-free encoding to the input in conjunction with the plain MD construction, then the (encoded) input to the MD construction may no longer satisfy the min entropy requirements from lemma E.1. Indeed, consider applying *Merkle-Damgård strengthening* to the input before the MD construction. In this case, the last block has no conditional entropy (since it is simply the input length). Nonetheless, in [11], Dodis et al. show that adding any fixed padding to an input that satisfies all min entropy requirements still gives a good randomness extractor! Similarly, we cannot say much about a general prefix-free encoding since it might change the input distribution in an arbitrary way. However, if we consider prepending input length to the input, then it still gives a good randomness extractor.

MD-THEN-CHOP CONSTRUCTION. Quite surprisingly, if we chop a sufficient number of output bits then one can prove randomness extraction properties of the resulting construction based on fewer assumption than lemma E.1. In particular, we can get rid of the requirement that the last input block has sufficient conditional min entropy.

Lemma E.2 ([11]) *Let $\mathsf{H}_{chop_s,r}$ be the MD-then-Chop construction defined over a family of random functions $\{h_r\}$, where the seed r is essentially the random IV used in the construction. Let $\mathcal{X}$ be the input distribution to H_{chop_s} (over bit strings with at most ℓ k-bit blocks). If $H_\infty(\mathcal{X}) = n{+}s{+}\log(\ell{+}1)$, then we get that $\mathbf{SD}(\mathsf{H}_{chop_s}(\mathcal{X}), U_{n-s}) \leq 2^{-s}$ where U_{n-s} is the uniform distribution on $(n-s)$-bit strings.*

NMAC/HMAC CONSTRUCTION. For the NMAC construction, it can be shown that if random and independent IV_1 and IV_2 are used in the two applications of the plain MD construction, then the resulting construction H_{nmac} is a good randomness extractor if the compression function represents a family of random functions. We can then restate lemma E.1 for the construction H_{nmac} as well, with the same exact security. However, it turns out that translating these results to the setting of the HMAC construction is not straightforward [11].